K-nearest neighbor algorithm and Perceptron algorithm for predicting stress level in couples based on their salaries.

-Convert the text data file to CSV as it is easy to manage data with CSV format.

-Import the data from the CSV file in form of a Matrix.

**For Perceptron:**

* -Import data and shuffle all the rows of it with random.shuffle() and create 2 matrices and store the first 320 rows as training data and next 79 rows as testing data.
* Initialize weights and set epochs in each repetition to 50 with total 10 repititions.
* So with a total number of 500 epochs we get the error rate as follows in fig 2.
* Here, we could see that for epochs=150, we get the optimized weights where error rate become lowest and after which it starts increasing steeply which means at 150 epochs we get the local minima of the cost function.
* So we get the performance metrics on training and test data for epoch=150 as in fig 3 and fig 4.

**For KNN:**

* Add 2 columns in the initial data matrix. One to note the distances and other to note predicted class.
* Iterate over the data to calculate the distance of each point with all other points and sort that matrix and save it in other matrix.
* If we are considering ’N’ neighbors, then slice the new matrix with (N+1) top rows as those are the closest neighbors which include the test datapoint itself whose distance is 0 from itself.
* If there are 2 classes then classify all these neighbors as class 1 or class 2 and add their distances.
* (Augmentation explaination): Let say there are 5 neighbors (A,B,C,D,E) of a point and out of which (A,B ) are class 1 and (C,D,E ) are class 2.
* Then add the distances and find mean of dist(class1)= (dist(A) +dist(B)) /2.
* Then add the distances of dist(class2) = (dist(C) +dist(D) +dist(E))/3
* Now compare dist(class1) and dist(class2) and whichever is low, assign the considered datapoint the respective class.
* We have selected to take the mean of the distance as lets say if there are 5 neighbors out of which 4 belongs to class 1 and are very near to the test datapoint but their aggregate distance will sum up greater than the 5th neighbor’s distance from datapoint.
* We have iterated the above setting for different values of neighbors from 2 to 25 and got the best results for **neighbors=2** which is visible from below Fig 1.
* We get all the performance metrics optimal for the above setting with neighbors as 2 , so we have selected the nearest neighbors as 2 with classwise mean weight augmentation.

**Review:**

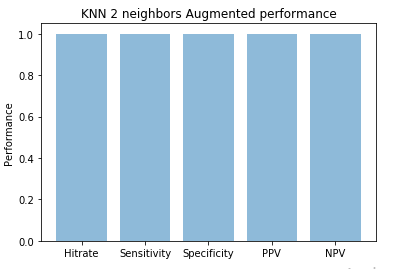
Here we could see from the bar graph of all the performance metrics (fig 1, fig3, fig4) that, all the positive metrics such as Hitrate, Sensitivity, Specificity, PPV, NPV are as high as 100% for KNN with mean weight augmentation while for Perceptron its around 87% on testing data.

Though the KNN is a lazy classifier, the difference of 13% on performance metrics is huge and hence we select the KNN classifier with mean weight augmentation(2-neighbors) for our dataset.

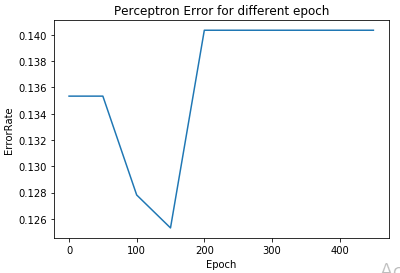
Moreover considering only 2 neighbors will increase the efficiency of the computation and a non-linearly separable data could be classified with KNN better than the perceptron as shown in the fig. 5 and fig 6.

**KNN**

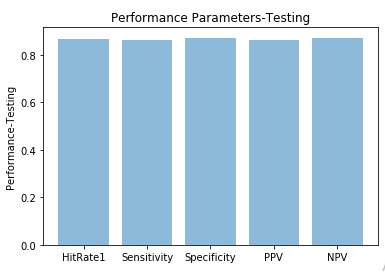
**Fig.1**



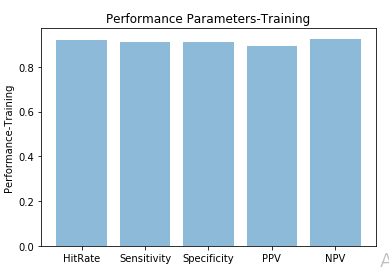
**Perceptron**

**Fig 2**

**Fig.3.**

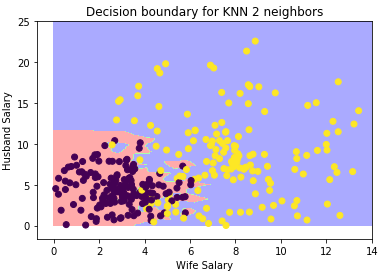


**Fig.4.**

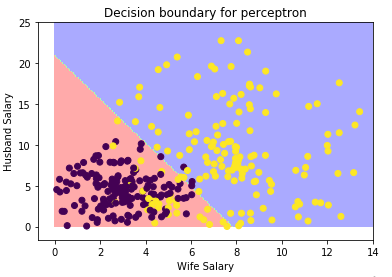


**Fig.5 Decision Boundary for KNN**

**(Red=Not stressed, Blue=Stressed)**



**Fig 6. Decision Boundary for perceptron.**



-